



**Konzept und Ansatz einer Wertschöpfungskette für die Erkennung
und Bereitstellung neuer Fahrumfänge intelligenter Fahrzeuge**

Forschungsseminar zur gleichnamigen Bachelorarbeit

An der
Carl von Ossietzky Universität Oldenburg
Studiengang Wirtschaftsinformatik

Vorgelegt von Linus Hestermeyer
linus.hestermeyer@gmail.com
Matr.Nr.: 4087097

Erstprüfer: Prof. Dr. Frank Köster
Zweitprüfer: Dipl.-Inform. Gerald Sauter

Oldenburg, den February 1, 2020

Inhaltsverzeichnis

1 Einführung	1
2 Bedarfserkennung von Software	2
2.1 Relevante Systeme eines autonomen Fahrzeugs in der Bedarfserkennung	2
2.2 Eine geeignete Kommunikationsgrundlage: OpenSCENARIO	3
2.3 Software-User-Pattern-Recognizer [SUPR] im Rahmen der Bedarfserkennung	5
2.3.1 1. Gezielte Suchanfragen	5
2.3.2 2. Ortsbezogene Erkennung eines Softwarebedarfs	6
3 Bereitstellung von Software	7
3.1 OTA-Aktualisierungen mit UPTANE	8
3.2 Anpassung von UPTANE Architektur	9
3.3 SUPR im Rahmen der Softwarebereitstellung	11
3.3.1 SUPR im Director Repository	11
3.3.2 SUPR im Fahrzeug	12
3.4 Die Mensch-Maschine-Schnittstelle	16
4 Technologie- und Methodik-Scouting	17
4.1 Arbeitsmethodik: User-Centered-Design	17
4.1.1 Personas	18
4.2 Tech-Scouting	20
5 Ausblick und erstes Konzept der praktischen Umsetzung	22

1 Einführung

"Durch den Einsatz von Elektrik und Elektronik ist das Automobil in den vergangenen Jahrzehnten stark geprägt worden, und die „Intelligenz“ in den Subsystemen hat exponentiell zugenommen." (Vgl. [4, S. 1]) Mit zunehmend mehr intelligenten Fahrassistenten in modernen Fahrzeugen ist es Absehbar, dass bereits in naher Zukunft Fahrzeuge die Stufe 3 bzw. teilweise Stufe 4 des Autonomen Fahrens erreichen. Ein Fahrzeug der Stufe 3 ist dazu in der Lage dazu, die Längs- und Querlenkung in bestimmten Anwendungsfällen selber übernehmen und diese so sicher zu durchfahren. Hierbei wäre allerdings noch ein Insasse notwendig, der in einem Notfall das Steuer übernehmen kann. In Stufe 4 Fahrzeugen kann das Fahrzeug die komplette Fahraufgabe in bestimmten Anwendungsfällen übernehmen. [16, S. 14]

Erst Stufe 5 Fahrzeuge werden "volumfänglich auf allen Straßentypen, in allen Geschwindigkeitsbereichen und unter allen Umfeldbedingungen die Fahraufgabe vollständig allein durchführen. Wann dieser Automatisierungsgrad erreicht sein wird, kann heute noch nicht benannt werden." (Vgl. [16, S. 14]). Aufgrund dessen ist es wichtig, dass Stufe 3 und Stufe 4 Fahrzeuge in Zukunft mehr Anwendungsfälle abdecken können. Hierzu sollen diese über eine kabellose Schnittstelle orts- und zeitunabhängig Softwarepakete herunterladen können, welche das Spektrum der autonomem Fahrfunktionen des Fahrzeugs zweckgebunden erweitert. Angenommen Sie planen mit ihrem neuen Fahrzeug eine Autofahrt nach England. Spätestens ab dem Ende des Eurotunnels wäre es für das Fahrzeug nicht mehr möglich selbstständig zu fahren, da dort Linksverkehr herrscht. Das Auto soll automatisch erkennen können, dass es ab einem bestimmten Punkt nicht mehr selbstständig fahren kann. In Folge dessen soll es eine Software zum Kauf/Miete anbieten, welche es dem Auto ermöglicht, am Linksverkehr teilnehmen zu können. Die Halter können ihr Fahrzeug dementsprechend zunehmend autonom fahren lassen, was den Marktwert des Autos nach dem Kauf steigern kann. Die Fahrzeughalter sollen bei Kauf von Software vom System unterstützt werden in Form von Vorschlägen für neue Software, die den Bedarf des Fahrers abdeckt. Diese Vorschläge sollen zu Zeitpunkten erfolgen, in denen der Verkauf einer bestimmten Software möglichst wahrscheinlich ist. Mittels dessen wird zudem unterbunden, dass der Nutzer von zu vielen Benachrichtigungen überfordert wird.

Damit Anwendungsfälle rasch abgedeckt werden können, bedarf es einem großen Spektrum an Software, die eben diese abdeckt. Um dies schnellstmöglich bewerkstelligen zu können ist es erforderlich, dass die Entwicklung von Softwarepakete durch Zulieferer geschehen, welche sich explizit auf diesen Markt fokussieren. Der hierdurch entstehende Seitenmarkt für die Entwicklung von Fahrfunktionssoftware kann somit schnell wachsen und sich als Teil in der Automobilindustrie etablieren. Durch autonome Fahrfunktionen werden Autos in der Regel schonender gefahren als vom Menschen, weshalb die Lebenszeit von Autos voraussichtlich verlängert wird. [16, (S. 16)] Ericssons Juergen Daunis sagt diesbezüglich, dass die meisten Analytiker und Führungskräfte der Meinung sind, dass der Umsatz dieser neu

entstehenden Seitenmärkte weiter steigen wird und dass das traditionelle Geschäftsmodell an dem wirtschaftlichen Maximum seiner Existenz ist. [6]

Die in dieser Arbeit erfassten Gliederungspunkte dienen als Grundlage für die gleichnamige Bachelorarbeit und sind im Wesentlichen als Literaturrecherche und Grundlagenerarbeitung zu verstehen. Es werden einzelne Bausteine der in der Bachelorarbeit zu erstellenden Wertschöpfungskette erstmalig benannt und konkretisiert. In Kapitel zwei wird dargestellt, wie das Auto einen Softwarebedarf erkennt, wie es einen Server hierüber informiert und wie dieser Server die richtige Software sucht. In Kapitel Drei wird zum einen eine sichere Architektur für kabellose Aktualisierungen erarbeitet. Des weiteren wird erläutert, wie Software verkauft wird und es werden Richtlinien für die Mensch-Maschine-Schnittstelle erarbeitet, auf welcher ein Angebot letzten Endes angezeigt werden soll.

2 Bedarfserkennung von Software

Im Rahmen der Bedarfserkennung steht die Beschreibung der eigenen Umgebung von Fahrzeugen im Mittelpunkt. Damit die Softwarehersteller den Bedarf von Fahrzeugen abdecken können, muss ihnen die Umgebung des Fahrzeugs zu dem Zeitpunkt der Bedarfserkennung bekannt sein. Um zu verstehen, wie ein Fahrzeug seine Umwelt eigens beschreiben kann, wird im Folgenden die Architektur eines autonomen Fahrzeugs vorgestellt und erläutert, welche Bestandteile dessen für die Bedarfserkennung relevant sind.

2.1 Relevante Systeme eines autonomen Fahrzeugs in der Bedarfserkennung

Um die Suche nach Software zu erleichtern, benötigt die Suche auf dem Server gewisse Daten als Input. Damit die Situation, in welcher das Fahrzeug nicht selbstständig fahren kann, durch eine Software abgedeckt werden kann ist es wichtig eine Beschreibung der Umwelt des Autos zu dem Zeitpunkt der Bedarfserkennung zu erstellen. Durch die Fusion aufgenommener Kamera-, Ultraschall- und Radardaten kann die Umwelt beziehungsweise die Umgebung des Autos in einem Datenformat wie dem von der ASAM.¹ definierten Standard "OpenSCENARIO" [8] dargestellt werden. Dieser wird in Kapitel 2.2 vorgestellt. Im Rahmen der Suche nach möglicherweise nötiger Software spielt die Routen- und Bewegungsplanung des Autos eine große Rolle. Zum einen kann das Auto bei der Eingabe einer neuen Route diese nach Gegenden absuchen in denen oft neue Software benötigt wird. Es kann infolgedessen diese dem Fahrer bereits vor Fahrtbeginn vorschlagen und somit die Bequemlichkeit der Fahrt sicherstellen. Zum anderen soll das Auto Muster im Fahrtenverlauf erkennen, um so bei der Erkennung eines Softwarebedarfs auf einer dieser Strecken eben diese Software zum Kauf vorzuschlagen.

Abbildung 1 zeigt die Architektur eines Autonomen Fahrzeugs nach Jeff Schneider. Sie

¹ <https://www.asam.net/standards/detail/opencENARIO/>

verdeutlicht, welche Bestandteile ein Autonomes Fahrzeug besitzt um mit Hilfe dieser selbstständig zu Fahren. Die aufgenommenen *Sensordaten* leiten Informationen an die *Karten- & Positionsverfolgung* weiter. Durch einen Merge (Zusammenführung) dieser Daten kann das Fahrzeug die eigene Umwelt *wahrnehmen* und mit Hilfe der dynamischen Inhalte der Welt (zB. Geschwindigkeiten) *Vorhersagen* für den Verkehr stellen. Als Bündel stellen sie die Bewegungsplanung dar.

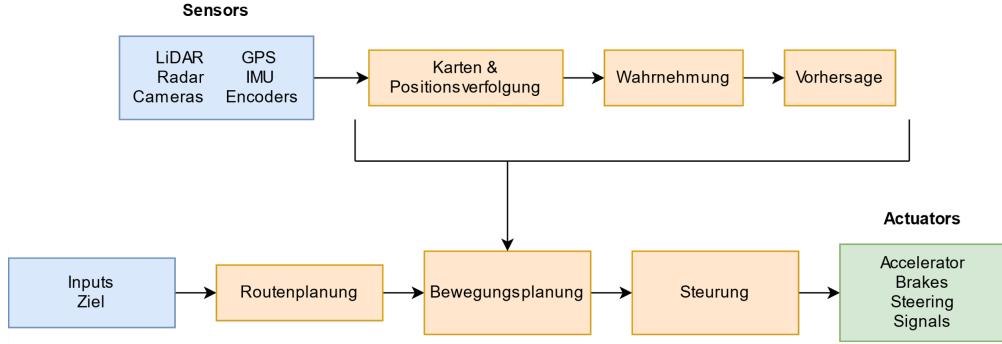


Figure 1: Architektur AV nach Jeff Schneider (CMU) [5]

Der Fahrer gibt dem Auto initial ein *Ziel*, mit Hilfe dessen das Fahrzeug die zu fahrende Route berechnet. Die Bewegungssteuerung gibt das Wahrgenommene und Vorhergesagte weiter an die Steuerung welche letztendlich entscheidet, was die einzelnen Aktuatoren machen. Damit der Bedarf einer Softwarelücke festgestellt werden kann, müssen die Systembausteine "Sensoren", "Karten & Positionsverfolgung", "Wahrnehmung" und "Vorhersage" angeknüpft werden um so herauszufinden **wann** das Fahrzeug nicht mehr selbstständig fahren kann. Wurde dieser Moment festgestellt, wird die Übergabe der Fahraufgabe initiiert und zeitgleich auch die Suche nach Software gestartet.

Neben den wichtigen Bausteinen der Bewegungsplanung kann auch anhand der Routenplanung von Fahrzeugen ein Softwarebedarf untersucht werden (Siehe 2.3). Zunächst wird eine Möglichkeit dargestellt, wie die Daten der Bewegungsplanung in einer geeigneten Form gespeichert und versendet werden können.

2.2 Eine geeignete Kommunikationsgrundlage: OpenSCENARIO

Der Inhalt dieses Kapitels basiert auf Inhalten der Projektwebseiten des OpenSCENARIO Standards. [8].

OpenSCENARIO ist ein XML-basiertes Dateiformat zur Beschreibung aller statischen (*Gegenstände, Hindernisse, etc.*) und dynamischen (*Geschwindigkeiten, Bewegungsrichtungen, etc.*) Inhalte der Umwelt eines Fahrzeugs. "Der primäre Anwendungsfall von OpenSCENARIO ist es komplexe, synchrone Manöver zu beschreiben, welche mehrere Entitäten wie Fahrzeuge, Fußgänger und andere Verkehrsteilnehmer betreffen." (Vgl. [8, asam.net])

Abbildung 2 zeigt einen Ausschnitt einer OpenSCENARIO-Datei. Für ein Szenario ist

immer das **Straßennetzwerk** (*RoadNetwork*) sowie die beinhalteten **Entitäten** (*Entities*) festzulegen. Das eigentliche Szenario ist innerhalb eines **Storyboards** dargestellt und unterteilt sich in einzelne **Storys** (Siehe Abbildung 2). Alle zuvor definierten Entitäten erhalten eine initiale **Action**, welche das initiale Handeln der Entität festlegt (Siehe <Init>-Block).

Eine einzelne Story ist immer einer einzelnen Entität zuzuordnen. Eine Story wiederum ist in "Acts" aufgeteilt, welcher eine Sammlung an "Sequenzen" beinhalten kann. Jedes dieser Elemente kann mit einer "Condition" versehen werden. Wird die "Condition" (Bedingung) erfüllt, wird der jeweilige Story-/Act- oder Sequenz-Block ausgeführt.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <OpenSCENARIO>
3
4  <FileHeader revMajor="0" revMinor="9" date="2017-02-24T10:00:00"
5    description="Sample Scenario - Overtaker" author="Andreas Biehn"/>
6
7  <ParameterDeclaration/>
8
9  <Catalogs>
38
39  <RoadNetwork>
40    <Logics filepath="Databases/PEGASUS/PEGASUS_A01.xodr"/>
41    <SceneGraph filepath="Databases/PEGASUS/PEGASUS_A01.opt.osgb"/>
42  </RoadNetwork>
43
44  <Entities>
45    <Object name="Ego">
51    <Object name="A1">
57  </Entities>
58
59  <Storyboard>
60    <Init>
61      <Actions>
62        <Private object="Ego">
79        <Private object="A1">
96        </Actions>
97      </Init>
98      <Story name="MyStory" owner="A1">
99        <Act name="MyAct">
100       <Sequence name="MySequence" numberOfExecutions="1">
175       <Conditions>
186       </Act>
187     </Story>
188   <End>
189   </End>
190 </Storyboard>
191
192 </OpenSCENARIO>
```

Figure 2: Ausschnitt einer OpenSENARIO Datei von Andreas Biehn [8, (Downloads)]

OpenScenario ist aus mehreren Gründen gut geeignet um als Kommunikationsgrundlage zwischen Auto und Server zu fungieren. Zum einen handelt es sich dabei um eine Open-sourcelösung, wodurch jeder Entwickler Weltweit selbstständig mit diesem Arbeiten kann. Zum anderen existiert wegen dem XML-basierten Dateiformat eine gute Lesbarkeit und Regelmäßigkeit Abfolge der Szenarien. Ein Szenario kann aufgrund dieser Regelmäßigkeit einfach und schnell erstellt werden. Der letzte und größte Vorteil von OpenScenario ist es, dass man ein Szenario in dem Opensource Simulator "Carla" abspielen kann.

Der im Folgenden vorgestellte Software-User-Pattern-Recognizer (SUPR) nutzt das OpenScenario-format zur Suche nach Software. Welche Suchvarianten es gibt und weitere Aufgaben des SUPR werden hier vorgestellt.

2.3 Software-User-Pattern-Recognizer [SUPR] im Rahmen der Bedarfserkennung

Das in diesem Abschnitt vorgestellte Konzept des Software-User-Pattern-Recognizer (kurz: SUPR) ist ein eigens ausgearbeiteter Baustein, welcher die Bedarfserkennung für intelligente Fahrzeuge beschleunigen und vereinfachen soll. Die Aufgaben des SUPR lassen sich in Aspekte der Bedarfserkennung sowie der Bereitstellung aufteilen, weshalb der SUPR in Kapitel 3.3 wieder aufgegriffen wird.

Eine Aufgabe des SUPR im Rahmen der Bedarfserkennung ist es, auf Anfrage hin eine Suche nach passenden Softwarepaketen durchzuführen (**1.**). Die zweite Aufgabe ist die regelmäßige Suche nach Software in der Umwelt/Region des Autos bzw. die Suche nach Softwarepaketen entlang zu Fahrender Strecken (**2.**). Das Maß der Rechenleistung auf Seiten des Servers ist indes höher als auf dem Fahrzeug. Eine mögliche Gegenüberstellung von Rechenleistung und Sendeleistung des Servers und des Autos kann zu behebende Defizite der Architektur aufdecken - die Optimierung des Systems ist jedoch nicht Teil des Forschungsseminars.

Der Server, auf welchem die Suche stattfindet, benötigt zwei Datenbanken: die eine enthält sämtliche Softwarepakete für intelligente Fahrzeuge, die andere eine große Menge an OpenScenario Dateien, welche zusätzlich Fremdschlüsselverweise auf ein oder mehrere Softwarepakete bereitstellen. Nur wenn diese Bedingung erfüllt ist, kann eine Suche durchgeführt werden.

2.3.1 1. Gezielte Suchanfragen

Gerät ein intelligentes Fahrzeug im Laufe seiner Fahrt in eine ihm unbekannte Situation, kann es diese mit Hilfe von OpenSCENARIO beschreiben. Das vom Auto erstellte Szenario wird an den Server geschickt, welcher dieses mit den auf der Datenbank liegenden Szenarien abgleicht. Ist die Suche abgeschlossen, wird eine Fallunterscheidung zwischen den folgenden Optionen getätigert:

A Es wird kein passendes Softwarepaket gefunden

Entweder existiert zu dem übergebenen Szenario keine Software oder möglicherweise

wurde die im Szenario dargestellte Situation noch nicht zu einer Software gemapped. Dies sollte von speziell hierfür angestellten Arbeitnehmern überprüft werden, um so Dopplungen in der Softwareentwicklung zu vermeiden.

B Es werden (mehrere) passende Softwarepakete gefunden.

In diesem Fall gilt zu entschieden, welches Softwarepakete die besten Auswirkungen auf die Performance des Autos hat. Hierzu ist das Heranziehen diverser Kennzahlen ratsam um somit die Performance der unterschiedliches Softwares untereinander zu vergleichen und eine fundierte Entscheidung treffen zu können. Eine beispielhafte Ausarbeitung dieser Entscheidungstreffung stellt Niklas Stelter in seiner Bachelorarbeit vor [21]. Am Ende einer Suche soll entweder eine einzelne oder keine Software vorgeschlagen werden.

Neben der vom Fahrzeug ausgehenden Suche nach einer bestimmten Software, kann dieses auch Vorschläge für Software von einem Server erhalten, welcher dauerhaft die Umwelt intelligenter Fahrzeuge analysiert.

2.3.2 2. Ortsbezogene Erkennung eines Softwarebedarfs

Neben der Suche nach einer bestimmten Software soll der SUPR auch Suchen in der Heimatregion des Autos durchführen sowie auf zu Fahrenden Strecken. Diese Suchen basieren nicht auf OpenScenario-Dateien sondern auf Basis der Routenplanung[A] (Siehe Abbildung 1) und der Fahrtenhistorie des Fahrzeugs[B].

A: Suche auf Basis der Routenplanung

Wird vom Fahrer eine nicht oft oder noch gar nicht zurückgelegte Strecke vorgegeben, so soll der SUPR entlang der zu fahrenden Strecke häufig auftretenden Softwarebedarf identifizieren und dem Fahrer vor oder kurz nach Antritt der Reise diese Softwarevorschlägen. Daraus ergibt sich die Anforderung, zu jedem sich in der Datenbank befindlichen Open-Scenario auch die geographische Position der Bedarfsentdeckung zu speichern. In Abbildung 3 ist folgende Situation dargestellt: Unser Auto hat als Ziel im Navigationssystem "Braunschweig" erhalten und befindet sich aktuell auf der A2. Die roten Boxen stellen den möglichen Bereich dar, in welchem der Server die Umweltanalyse betreibt. Wird eine Software entdeckt, überprüft das System die Relevanz für das eigene Fahrzeug und schlägt sie je nach dem vor oder nicht.

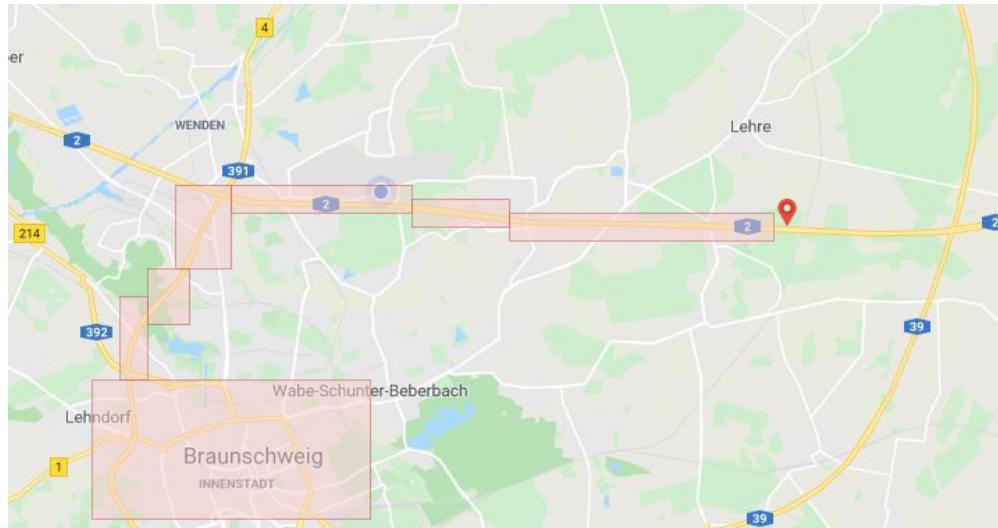


Figure 3: Umweltanalyse

B: Suche auf Basis der Fahrtenhistorie

Hierbei werden Muster in den zurückgelegten Strecken des Fahrers gesucht, um so dessen meist gefahrene Strecken zu identifizieren. Das Auto soll abschätzen können, wann der Fahrer welche Strecke zurücklegen wird. Vor dem jeweiligen Fahrtantritt soll die Suche nach Software identisch zu Variante A durchgeführt worden sein. Der Unterschied ist, dass die Suche hierbei selbstständig erfolgen soll, was ein besseres Nutzungserlebnis für den Fahrer schafft.

Ist die Suche (*egal ob gezielte Suche oder Umweltanalyse*) abgeschlossen, kann die Software dem Fahrer zum Kauf vorgeschlagen und bei Bestätigung für das Auto bereitgestellt werden. Das nächste Kapitel verdeutlicht den Umfang der Bereitstellung.

3 Bereitstellung von Software

Wurde der Bedarf einer neuen Software vom Fahrzeug festgestellt, muss diese im folgenden bereitgestellt werden. Zuvor muss der Softwarezulieferer die Sicherheit von Softwarepaketen verifizieren und eine sichere Kommunikation zwischen Servern und Fahrzeugen herstellen. Der Server muss alle vorhandenen Softwarepakete verwalten und auf Anfrage das am besten geeignete Softwarepaket für das jeweilige Fahrzeug identifizieren und ein Angebot an dieses schicken. Das Fahrzeug muss die eingegangenen Softwareangebote über die Mensch-Maschine-Schnittstelle zu einem Zeitpunkt anbieten, an dem die Kaufbereitschaft des Fahrers möglichst hoch ist. Hierzu bedarf es einem Softwaremanagement, welches die Angebote des Servers verarbeitet. Da die Grundvoraussetzung der Wertschöpfungskette eine sichere Kommunikation zwischen Auto und Softwarelieferanten ist, wird zunächst eine mögliche Architektur zur sicheren Kommunikation vorgestellt.

3.1 OTA-Aktualisierungen mit UPTANE

Der im folgenden vorgestellte Standard "UPTANE", stellt eine Architektur und Vorgehensweise für die sichere Kommunikation zwischen einem Server und einem Auto vor. Hierdurch können Software-Aktualisierungen an intelligente Fahrzeuge sicher verteilt werden [7]. Erste Schritte hierzu wurden 2010 getätig, als Justin Samuel, Nick Mathewson, Roger Dingledine und Justin Cappos "*The Update Framework*" (*TUF*) entwickelten. Dieses bildet den Grundbaustein für den späteren "**UPTANE**"-Standard. Mittlerweile ist es Teil des "Automotive Grade Linux Projekts" und somit Teil der "Linux Foundation". Uptane stellt mittels einer Mehrschichtenarchitektur sicher, dass im Rahmen von Aktualisierungen keine Schädlingssoftware auf ein Auto gelangt. Zunächst wird die Architektur von Uptane dargestellt (Abb. 4).

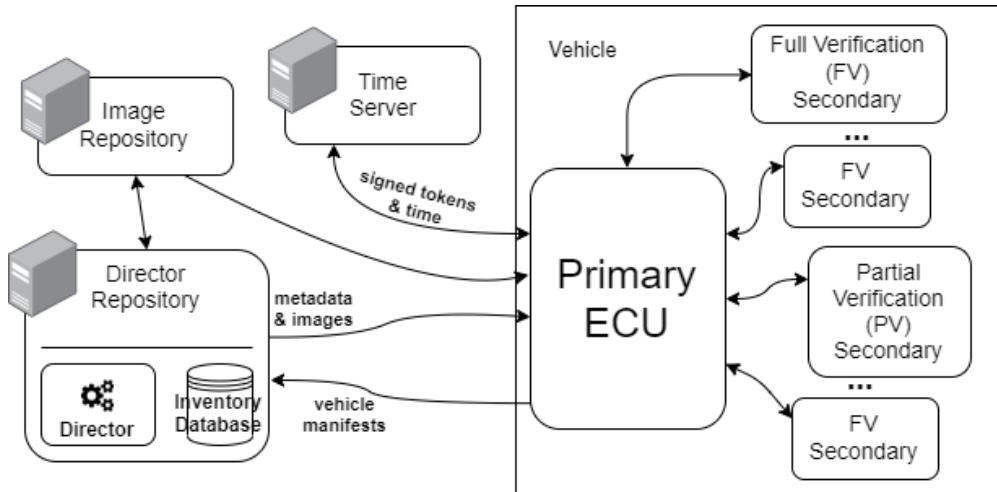


Figure 4: Architektur Design UPTANE [7]

Es ist vorab wichtig zu erwähnen, dass Uptane lediglich der Standard ist und keine offizielle Implementierung bereitstellt. Beispielhafte Implementierungen wären aktualizr¹, rust-tuf², Notary³ oder die OTA Community Edition⁴. Kommerziell entwickelte Systeme werden bereitgestellt von HERE Technologies⁵ sowie von Airbiquity⁶.

Die rechte Seite des Bildes stellt das Fahrzeug dar, die Elemente zur linken die Repositories. Diese Repositories sind Server und sie haben alle eine eigene wichtige Aufgabe. Der Time-Server ist dafür da, um ECUs über die aktuelle Zeit zu informieren, da viele ECUs keine Uhr haben [7]. Das Image Repository speichert jedes derzeit vom Lieferanten verteilte Image zusammen mit den Meta-Daten, welche zur Authentizität benötigt werden. Es nutzt

¹ <https://github.com/advancedtelematic/aktualizr>

² <https://github.com/heartsucker/rust-tuf>

³ <https://github.com/theupdateframework/notary>

⁴ <https://github.com/advancedtelematic/ota-community-edition/>

⁵ <https://www.here.com/products/automotive/ota-technology>

⁶ <https://www.airbiquity.com/product-offerings/software-and-data-management>

Offline-Schlüssel um alle Metadaten zu "unterschreiben" bzw. zu verifizieren, was einen hohen Sicherheitsvorteil darstellt. Das Director Repository entscheidet abhängig von den übergebenen Informationen des Autos genau, welche Images an die ECUs verteilt werden müssen. Ein Auto hat mehrere ECUs, welche sich in Speicherplatzgröße, Stromverbrauch und Aufgabenbereich unterscheiden. Die *Primary ECU* verwaltet und steuert die Installationen auf den anderen ECUs.

In dem ersten Schritt einer Aktualisierung schickt das Fahrzeug sein Manifest an das Director Repository. Dieses enthält Informationen darüber, welche Images aktuell auf dem Auto installiert sind. Das Director Repository entscheidet anhand dessen, welche Software auf dem Fahrzeug installiert/aktualisiert werden soll. Die Metadaten und die neuen Images werden zurück an die ECU geschickt. Hier findet zunächst eine Verifikation der neuen Images statt, bevor sie anschließend bei erfolgreichem Test installiert werden. Die Verifikation der ECUs kann entweder vollständig oder teilweise erfolgen. **Vollständig** heißt in diesem Kontext, dass die Größe der Software und die Hashes, welche die ECU über die Metadaten vom Director Repository erhält, identisch sind zu denen der Metadaten die vom Image Repository zur Verfügung gestellt werden. Bei der **teilweisen** Verifikation muss lediglich die Signatur der Metadaten des Director Repositories mit der Signatur der Metadaten vom Image Repository übereinstimmen.

3.2 Anpassung von UPTANE Architektur

Damit die durch UPTANE bereitgestellte Architektur die Erkennung und Bereitstellung neuer Fahrumfänge unterstützt, muss diese dementsprechend angepasst werden, damit neben dem Aktualisieren bereits installierter Software auch das Installieren neuer Software möglich ist. Hierzu wird die Architektur von UPTANE erweitert und angepasst.

Damit das Director Repository nicht nur bestimmen kann, welche Software eine Aktualisierung benötigt, sondern auch welche auf dem Fahrzeug installiert werden muss um den Bedarf zu decken, braucht es eine geeignete Kommunikationsgrundlage. Hierzu soll das zuvor vorgestellte OpenSCENARIO XML-Speicherformat verwendet werden. Um die Suche nach neuer Software für das Directory Repository zu ermöglichen, wird das in Abbildung 5 dargestellte "*Scenario Repository*" eingeführt. In diesem werden OpenSCENARIO-Dateien gespeichert, welche als Basis für die Suche nach Software dienen. Jede Datei hat hält mindestens **eine** Referenz in Form eines Schlüssels auf eine Software aus dem Image Repository.

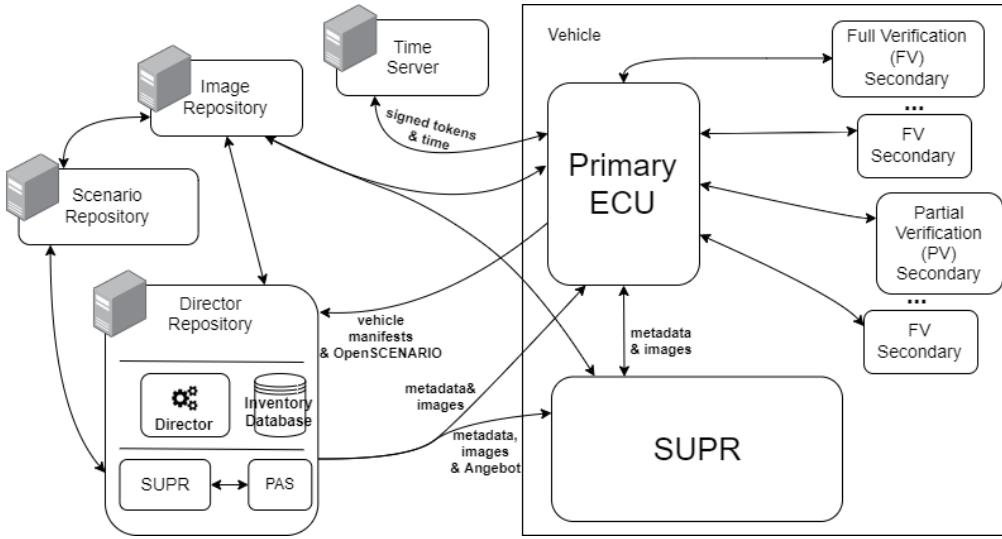


Figure 5: Angepasste Uptane Architektur

Dies bedeutet, dass die jeweilige Situation der OpenScenario-Datei von eben dieser Software bewältigbar ist. Erhält das Director Repository nun eine Anfrage für eine neue Software, wird diese anhand der mitgeschickten OpenSCENARIO-Dateien gesucht. Bei erfolgreicher Suche, soll diese in das Scenario Repository eingetragen werden. Die Suche führt der in Kapitel 2.3 eingeführte Software-User Pattern-Recognizer durch. In Kapitel 3.4 wird dieser um die Aspekte der Bereitstellung von Software erweitert.

Fahrzeughalter müssen neue Software kaufen, weshalb im Fall einer gefundenen Software zusätzlich zu den Metadaten und dem Softwarepaket (*Softwareimage*) noch eine Sammlung verkaufsbezogener Daten verschickt werden muss. Diese wird hier und im folgenden **Angebot** genannt. Ein Angebot muss alle möglichen Verkaufsarten (*Kauf, Miete, o.A.*) sowie dazu passende Preise beinhalten. Da die Preise und Verkaufsarten für Software je nach Automarke und -model variieren können sollen, wird ein Modul "pricing and sales", kurz PAS, zur Bestimmung des Preises und der Verkaufsart vorgeschlagen (*siehe Abbildung 5*). Das Director Repository ist nach wie vor dazu in der Lage, Softwareimages und Metadaten direkt an die Primäre Recheneinheit (ECU) zu schicken, wenn es sich nur um eine Aktualisierung bereits installierter Software handelt. Wird hingegen eine neue Software vorgeschlagen, so werden Image und Metadaten zusammen mit einem Angebot an den Software-User Pattern-Recognizer des Autos verschickt. Alternativ wird nur das Angebot verschickt um abzuwarten, ob der Kunde die Software tatsächlich haben will. Hierdurch würde das Image nicht unnötiger Weise verschickt werden, was unter anderem die Ressourcen schont. Der SUPR des Autos identifiziert unter stetiger Beobachtung des Fahrers, wann dieser einen Softwarevorschlag verarbeiten kann. Er stellt zum gegebenen Zeitpunkt Verbindung zu der Nutzeroberfläche des Wagens her, um so mit dem Fahrer zu kommunizieren und den Kaufvorgang abzuschließen (*Ob erfolgreich oder nicht ist hierbei egal*). Die Funktionsweisen und Aufgaben des SUPR im Auto unterscheiden sich von denen

des SUPR im Director Repository. Aufgrund dessen ist es wichtig eine Abgrenzung und Spezifizierung dieser vorzunehmen.

3.3 SUPR im Rahmen der Softwarebereitstellung

Das folgende komplettiert den in Kapitel 2.3 vorgestellten Software-User Pattern Recognizer (*SUPR*). Wie im ersten Teil wird auch im Rahmen der Bereitstellungskomponente des SUPR zwischen den Aufgaben auf Server und denen im Fahrzeug unterschieden.

3.3.1 SUPR im Director Repository

Die unterschiedlichen Suchvarianten welche der SUPR im Rahmen der Bedarfserkennung bereits auf dem Server ausführt, wurden in Kapitel 2.3 spezifiziert. Im Rahmen der Software Bereitstellung ist es die wesentliche Aufgabe, das Angebot und alle möglichen Preise festzulegen. Die verschiedenen Preise ergeben sich aus Kombination der einzelnen **Einflussfaktoren**.

Ein Angebot, welches vom Director Repository (siehe 3.2) an das SUPR des Fahrzeugs geschickt wird, spezifiziert die Preise für Software in Abhängigkeit der **Kaufart**, der **vo-
raussichtlichen Laufzeit** des Mietverhältnisses (bei Miete), der Menge der Software welche der Fahrzeughalter in dem Moment akquirieren möchte und den eigentlichen Herstellungskosten der Software. Die folgende Tabelle definiert die Werte, welche die einzelnen Einflussfaktoren annehmen können.

Wert	Beschreibung
Kaufart	
Kauf	Der Fahrer akquiriert die Software dauerhaft. Eine Reklamation ist möglich, wenn diese ihre Aufgabe nicht korrekt ausführen würde.
Leihe	Der Fahrer legt einen Zeitraum(<i>von .. bis</i>) fest, für welchen er diese Software akquirieren möchte. Für diesen bezahlt er im Voraus und die SW wird bei Ablauf des Zeitraums deinstalliert.
Miete/Abo	Der Fahrer wählt die Dauer einer Mietphase, an welche sich der Preis anpasst. Das Mietverhältnis wird nach Ablauf dieses Zeitraums aufrecht erhalten. Das heißt es entstehen laufend Kosten bis es vom Fahrer beendet wird.

Zeitraum/Dauer	
Permanent	Der Zeitraum ist nur dauerhaft, wenn die Software gekauft wird. Hierbei tritt der höchste Preis auf.
Lang	Sechs Monate oder mehr. Auf einen Tag heruntergerechnet der günstigste Miet- oder Leihpreis.
Mittel	Sechs Wochen bis zu sechs Monate. Auf einen Tag heruntergerechnet ein wenig teurer als der des Langen.
Kurz	Acht Tage bis sechs Wochen. Auf einen Tag heruntergerechnet ein wenig teurer als der des Mittleren.
Einmalig	Ein bis sieben Tage. Auf einen Tag heruntergerechnet das teuerste Angebot.

Menge	
Einzeln	Es wird nur eine Software zu dem Zeitpunkt angeboten. Hat keinen Einfluss auf den Preis.
Mehrere	Software wird im Bundle gekauft, was sich positiv auf den Preis auswirkt. Ein Bundle muss vom SUPR im Fahrzeug erstellt werden.
Flottenkauf	Es wird Software für mehrere Fahrzeuge gekauft. Dies kann für Unternehmen Sinnvoll sein, die Dienstwagen anbieten oder auch Autovermietungen etc.

Weitere Einflüsse auf den Preis	
Herstellungskosten	Die Kosten die im Laufe der (Weiter-)Entwicklung und der Wartung dieser Software entstanden sind.
Beliebtheit	Ist eine Software beliebt bzw. gefragt, steigt der Preis dieser. Die Beliebtheit kann entweder global gemessen werden oder regionsabhängig. Der Preisanstieg sollte dabei nicht zu hoch sein.
Mächtigkeit	Je mehr eine Software "kann", desto teurer sollte diese sein.

Nachdem der SUPR des Director Repositorys eine den Bedarf deckende Software gefunden hat, muss er die Referenz auf die Datei an das PAS-Modul (*siehe Abbildung*) schicken. Das PAS-Modul soll anhand der Software, einiger Informationen zum Kaufverhalten der Fahrer des Fahrzeugs sowie dem generellen Bedarf der Software ein personalisiertes *Angebot* erstellen und anschließend an das Fahrzeug schicken.

3.3.2 SUPR im Fahrzeug

Neben dem Server-seitigen SUPR hat auch der SUPR des Autos wichtige Funktionen im Rahmen der Bereitstellung von Software. Ist das Angebot im Auto angekommen, identifiziert dieser einen geeigneten Verkaufszeitpunkt und stellt zu diesem das personalisierte Angebot über die Mensch-Maschine Schnittstelle (MMS) dar. Dies meint den Zeitpunkt, an dem der Verkauf oder die Leihgabe von Software möglichst wahrscheinlich ist. Hierzu ist eine Beobachtung des Fahrers nötig, um anhand von dessen **Eigenschaften** wie bspw. Mimik,

Gestik, Sprache oder dem Fahrverhalten das Stress-, Freude, oder Angstlevel zu deuten, aber auch um Müdigkeit oder Ablenkung feststellen zu können. Vor allem Faktoren wie "Stress, Freude und Angst beeinflussen unsere Meinung und Entscheidung." Vgl. [20, S.44]. So wird eine Software eher nicht gekauft, wenn der Fahrer in eine stressige Verkehrssituation überblicken muss, zum Beispiel wenn es dicht benebelt ist und man zur Zeit in einer unbekannten Gegend Auto fährt. Hingegen beeinflusst Freude einen eher dazu, einen Kauf zu tätigen. Angst ist für den Anwendungsfall des Verteilens neuer Fahrfunktionen divers zu deuten. Wenn der Fahrer Angst vor der Strecke hat die vor ihm liegt, muss der SUPR dies anders bewerten als Angst die Aufgrund anderer Einflüsse entsteht. Diese Beobachtung entspricht der dritten Richtlinie Mensch-zentrierter autonomer Fahrzeuge nach Lex Fridman. [17, S. 3] Nach der fünften Richtlinie Fridmans, soll ein Auto von dem Moment dem ersten Einstiegs des Fahrers auf diesen personalisiert werden. [17, S. 5] So ist es Sinnvoll, würde der SUPR persönliche Eigenschaften in seine Entscheidungsfindung zum geeigneten Verkaufszeitpunkt einfließen lassen.

Ein System eines Fahrzeugs soll laut Fridman Daten-orientiert sein [17, S. 3]. Das heißt, dass ein Fahrzeug aus den aufgenommenen Daten lernen soll. Fusioniert man die bei der Fahrerbeobachtung aufgenommene Daten mit Daten über den Verkaufserfolg, kann der SUPR aus seinen Handlungen lernen und sich so optimieren. Hierdurch werden mögliche den Kauf negativ beeinflussende Faktoren identifiziert (zB. Stress) und der Fahrer so besser kennengelernt. Die Kaufbereitschaft des Fahrers soll in Folge dessen möglichst oft korrekt eingeschätzt werden.

Ist der Zeitpunkt bzw. Zeitrahmen identifiziert, stellt der SUPR über die Mensch-Maschinen-Schnittstelle das personalisierte Angebot dar. Der dargestellte Inhalt wird maßgeblich beeinflusst von den Aspekten der Verkaufspräzesspsychologie. Um die wesentliche Aufgabe des SUPR (*Beobachtung & Kommunikation mit Fahrer, Vermarkten von Software*) Sinnvoll zu modellieren, werden die Einflussfaktoren für den dargestellten Inhalt der MMS anhand der Prinzipien der Verkaufspolitik nach Markus Reinke [18] erläutert.

Um einen Kunden vom Erwerb eines Produkts zu überzeugen, muss diesem etwas angeboten werden wodurch ein offenes Bedürfnis befriedigt. Dazu muss herausgefunden werden, welches Produkt geeignet ist, damit es so anschließend angeboten werden kann. Wie der SUPR die Bedürfnisse von Fahrern identifiziert ist in Kapitel 2.3 nachzulesen. Was bei dem Anbieten einer Software beachtet werden muss, wird in diesem Kapitel abgehandelt. Kunden sind unterteilbar in Plus-, Chancen- und Minus-Kunden. [18, S. 10ff.] Minus-Kunden werden ein Produkt von Beginn an nicht kaufen wollen, so gut der Erwerb auch gestaltet sein wird. Plus-Kunden werden ein Produkt voraussichtlich kaufen. Um die Chancen-Kunden von einem Erwerb zu überzeugen, werden im Absatz von Unternehmen diverse Prinzipien angewendet. Es kann also die Schlussfolgerung gezogen werden, dass so gut der SUPR seinen Fahrer auch kennt und wie gut Produkte den Bedarf auch abdecken,

der Kunde dennoch nicht immer eine Software erwerben wird. Um die Anzahl an Käufern möglichst hoch zu halten, soll der persönliche Nutzen den die SW für den Kunden hat dargestellt werden. So kann die Situation simuliert und dargestellt werden, in welcher das Fahrzeug den Softwarebedarf festgestellt hat.

Das Differenzprinzip [18, S. 19fff.]

Nach dem Differenzprinzip sollen unterbewusste Reize gesetzt werden, die zum Erwerb leiten. Zum Beispiel wird mit der teuersten Variante des Produkts geworben, damit der Kunde bei näherem betrachten des Angebots die tieferen Preise, also ein "kleineres Übel", entdeckt und sich darüber erfreut. Dabei soll nicht die Frage **ob** der Kunde bei einem kaufen möchte, sondern **was** der Kunden von einem kaufen will.

Stellt der SUPR das Angebot dar, soll eine Mietfunktion im Fokus stehen, zudem deutlich gemacht werden. Die Option "Software Kaufen" ist kleiner darzustellen zusammen mit dem jeweiligen Preis. Um dem Kunden die Möglichkeit zu lassen die Software nicht zu erwerben, muss ein dezenter Knopf in der Nutzeroberfläche sein, welcher den Angebotsprozess beendet. Durch die Unauffälligkeit kann der Kunde zum Kauf gelenkt werden bzw. zu der Frage **was** man kaufen möchte, nicht **ob**. Um einen unterbewussten Reiz zu setzen, kann eine Karte dargestellt werden wie häufig die Software in der Umgebung installiert wurde. Dies sollte allerdings nur geschehen, wenn die Nachfrage tatsächlich auffällig hoch ist. Hiermit würde auch der Nachahmungseffekt abgedeckt werden (*weiter unten*).

Das Do-ut-des-Prinzip [18, S. 33fff.]

Nach diesem Prinzip, wird "gegeben, damit du gibst". Der Kunde wird durch beispielsweise Gratis-Proben angelockt um zum Erwerb bewegt zu werden. Will er dennoch nichts kaufen, kann um Weiterempfehlung gebeten werden. Dieser Bitte wird der Kunde vrs. nachkommen, da er im Vorfeld etwas "gratis" erhalten hat. Die "*Zwei Schritte vor, einer Zurück Taktik*" hierbei anzuwenden ist sinnvoll. Das heißt, dass dem Kunden zuerst die teuersten Produkte gezeigt werden, um ihn anschließend auch hier mit niedrigen Preisen anderer Produkte beeindrucken und halten zu können. Es sollte ist hierbei auch nicht das Ziel das teuerste Produkt zu verteilen, sondern das eigentliche Ziel ist eines mit tieferem Preis.

Der SUPR kann dieses Prinzip anwenden, in dem er dem Kunden zu neu erworbener Software eine Test-Version einer anderen Software "schenkt", welche nach einem bestimmten Zeitraum (4 Wochen) abläuft. Diese Software sollte einen bestehenden Bedarf der Kunden abdecken.

Das Konsequenzprinzip

Konsequenz im Handeln wird in den meisten Kulturen als eine positive Charaktereigenschaft gewertet, strahlt Sicherheit aus. Zudem schafft es Vertrauen bei Kunden. Konsequenz kann unter anderem mittels wenn-dann-Fragen ausgestrahlt werden, also "Wenn wir Ihnen XY bieten, kaufen Sie dann?" Der Kunde würde bei Erfüllung seines Bedarfs eher kaufen. Und

da der von ihm benannte Bedarf abgedeckt ist, wird er so von einem frühen Commitment überzeugt. Es sollen die Wünsche und Prioritäten der Kunden erkannt werden und anschließend das Produkt mit Fokus auf diese beworben werden.

Dieses Prinzip sollte vor allem verwendet werden, um einen "Fuß in die Tür zu bekommen" und nicht um das teure Produkt zu verkaufen. Sinnvoll ist es, nach dem Motto "Kleinvieh macht auch Mist" zu verkaufen. Der SUPR tut dies zum einen indem er die Situation, in welcher der Bedarf festgestellt wurde, auf der MMS darstellt und den Kunden so erinnert. Wenn der Kunde noch gar keine oder nur wenig Software erworben hat, soll ihm immer die Möglichkeit einer kostenlose Probe-Woche geboten werden. Dies soll bis zu fünfmal geschehen, danach wird ohne weiteres keine Gratis Software mehr angeboten.

Das Nutzen des Nachahmungseffekts [18, S. 67fff.]

"Der Mensch ist grundsätzlich ein Gemeinschaftswesen und achtet daher sehr darauf, was andere von ihm denken, sowohl in der breiten Öffentlichkeit als auch in seinem engen persönlichen Umfeld." - Markus Reinke (Vgl.) [18, S. 67]

Dieses Verhalten, der sogenannte **Nachahmungseffekt**, kann durch das Einsetzen verschiedener Techniken provoziert werden. Bei der **Zeugenumlastung** lässt sich der Kunde bestätigen, dass ihr Unternehmen und Produkt "gut" ist indem er von anderen positives darüber in Erfahrung bringt. Eine Variante, wie diese vom SUPR angewendet werden kann, wurde zuvor bereits dargestellt. Bei der **Referenztechnik** werden die Stammkunden eines Unternehmens gebeten Empfehlungsschreiben zu erstellen mit welchen im folgenden geworben werden kann. Der SUPR kann hierzu ein Bewertungssystem einführen und Nutzer können die Bewertung anderer Fahrer lesen und sich so eine Meinung der Software einholen. Dabei ist es sinnvoll, einige vorgefertigte Antwort/Bewertungsmöglichkeiten selber zur Verfügung zu stellen. Eine dritte Technik ist das **Empfehlungsmarketing**, bei welchem Kunden gebeten werden uns an Freunde und Verwandte weiter zu empfehlen. In den zeiten von Social Media bietet es sich an, eine "Teilen"-Funktion zur Verfügung zu stellen. Der Fahrer soll Teilen können, wie viele Kilometer er in welcher Zeit zurückgelegt hat, wie viele davon autonom bewältigt wurden und welche Softwarepakete genutzt wurden.

Um eine hohe Resonanz herzustellen, ist es sinnvoll dem Kunden ähnlich zu sein, sympathisch zu wirken, Lob & Anerkennung zu verteilen als auch andere zu unterstützen. Da es ungewohnt ist, Lob oder anderes von einem Computer zu erhalten, der SUPR aber dennoch eine hohe Resonanz erzielen soll, sollte den Kunden eine Bezugsstelle gegeben werden. Eine Referenz aus der Wirtschaft hierzu stellt "Meet Olli" dar [2]. Olli ist das Maskottchen des UserInterfaces (UI) von "Local Motors" (LM). LM stellt einen autonom fahrenden Bus her, mit welchem die Insassen über eine UI interagieren können. Olli hat "nur" zwei Augen und einen Mund, stellt hierdurch aber einen vermenschlichten Bezugspunkt für den Fahrer und die Insassen dar und gewinnt so das Vertrauen der Insassen.

Darüber hinaus soll der Kunde nicht von den Angeboten "erschlagen" werden, sondern

der SUPR muss diese dosiert vorlegen. Wenn der Kunde kein Interesse hat, ist dies zu respektieren und er soll damit nicht weiter konfrontiert werden. Der SUPR soll nur die tatsächlich benötigte Software bewerben - ein einmalig aufgetretener Bedarf fällt da nicht drunter (*Die Ausnahme ist, der Kunde ist ein absoluter Plus-Kunde*). Es muss genügend Werbung betrieben werden damit möglichst viel Software verkauft wird. "Klassische" Werbung auf der MMS ist nicht Sinnvoll, sondern eher Werbung in Form von Gratis-Proben von Software-Paketen. Nutzt der Fahrer eine dieser Proben und die SW kommt zum Einsatz, kann der SUPR den Fahrer über die MMS darüber in Kenntnis setzen, wodurch er vor Augen geführt bekommt, dass er diese Software braucht.

Ein Auto wird allerdings nicht nur von einer Person gefahren, sondern möglicherweise von mehreren Familienmitgliedern, Mitarbeitern oder anderen Mietern im Falle von Carsharing. Das SUPR muss dazu in der Lage sein, diese einzelnen Personen auseinanderhalten zu können um keine falschen Entscheidungen zu treffen.

Wird dem SUPR ein Softwarevorschlag mit Angebot zugeschickt, soll dieser bis das Angebot angezeigt werden kann, diese Software schon anfangen herunterzuladen. Dadurch kann die Software, wenn sie erworben wird, bereits während der Fahrt installiert werden, was den Verkauf schneller und Nutzerfreundlicher gestaltet.

3.4 Die Mensch-Maschine-Schnittstelle

Die Mensch-Maschine-Schnittstelle ist das System, über welches der Fahrer und das Auto mit einander interagieren. Im Kontext intelligenter Fahrzeuge handelt es sich hierbei meist um einen Bildschirm, mitunter haben Fahrzeuge auch einen integrierten Sprachassistenten.

Das Bereitstellen neuer Fahrfunktionen schließt mit ein, dass ein Fahrzeug zum Zeitpunkt A selbstständig Fahren kann und zum Zeitpunkt B nicht. Zwischen diesen Zeitpunkten muss eine Übergabe der Fahraufgabe von dem Fahrzeug an den Fahrer erfolgen. Dabei soll der Fahrer des Fahrzeugs über die momentane Verkehrslage ausreichend in Kenntnis gesetzt werden, um während und nach der Übergabe der Fahraufgabe die Sicherheit zu wahren. Der Begriff ausreichend definiert sich dabei wie folgt:

**Die Inkenntnissetzung gilt als vollständig, wenn die
Mensch-Maschine-Schnittstelle alle Prinzipien von S. Debernard et Al. [19]
erfüllt.**

Das zu entwickelnde Display soll Nutzer-zentriert (User-Centered) entwickelt werden, um die Anforderungen möglichst vieler Nutzer erfüllen können und zusätzlich den Faktor der "Vermenschlichung" zu erhöhen. Die geschaffene transparente Straßenführung über das Display eines Fahrzeugs, soll zusätzlich durch Audio-Assistenten unterstützt werden. Die MMS wird so durch zu einem audiovisuellem System.

Die Schnittstelle soll sich sowohl visuell als auch auditiv an die im Fahrzeug sitzenden Personen anpassen. So kann zum Beispiel die Schriftgröße größer sein, wenn eine Person im

Rentenalter das Fahrzeug betritt oder ähnliches. Mögliche Features der Personalisierung (*z.B. größere Icons, dunkles Design*) werden mit Hilfe der Personas 4.1.1 in der Bachelorarbeit erarbeitet. Im generellen soll der Nutzer auf dem Display eine Simulation des eigenen Autos sehen zusammen mit zusätzlicher Information gemäß den Prinzipien nach Debernard et. Al. Das Design des User-Interfaces kann vom Fahrer angepasst werden, wobei die intuitive Bedienung sich nicht verschlechtern soll. Durch die Vermenschlichung des MMS, einer Transparenten Straßenführung, der Möglichkeit die Steuerung jederzeit zu übernehmen sowie der höflichen Kommunikation mit dem Fahrer, wird das Vertrauen des Fahrers zu dem Fahrzeug gefördert [2], was den Kauf von Software wahrscheinlicher macht.

4 Technologie- und Methodik-Scouting

Um auf Grundlage der dargestellten Rahmungen für die Bedarfserkennung und Bereitstellung neuer Fahrumfänge für intelligente Fahrzeuge einen Prototypen zu entwickeln, wird abschließend ein Technologie- und ein Methodik-Scouting durchgeführt werden. Das Kapitel der Arbeitsmethodik spezifiziert Den Zyklus des User-Centered-Designs und stellt dar, weshalb dieser für die Entwicklung des Prototypen die richtige ist. Das anschließende Technologie-Scouting stellt Entwicklungsumgebung, Frameworks und Programmiersprachen vor, welche in der Entwicklung genutzt werden.

4.1 Arbeitsmethodik: User-Centered-Design

Um die Nutzeroberfläche für die Zielgruppen ansprechend zu gestalten, wird bei der Erarbeitung dieser nach den Prinzipien des User-Centered Designs (*UCD*) gehandelt. Ins deutsche übersetzt bedeutet dies "Benutzerzentriertes Design bzw. Benutzerorientierte Gestaltung" eines Produkts. Es beschreibt einen Designprozess bzw. ein Entwicklungsverfahren, "auf welches der Endnutzer (Fahrer) schon von Anfang an Einfluss nimmt" (Vgl. [15, S. 763]). Der User wird in die Phasen der Entstehung einer Benutzungsschnittstelle integriert was zur Folge hat, "dass der Aufbau, die Inhalte und deren Form sowie das Design des Endproduktes maßgeblich von den Bedürfnissen, Erwartungen und dem Verständnis der User bestimmt wird" (Vgl. [1]). Des weiteren kann durch das Einbeziehen von Kunden die Qualität optimiert werden (Vgl. [12, S. 14]).

"Typisch für einen UCD-Prozess zum Entwerfen von Webanwendungen mit optimaler User Experience sind [jedoch] die Prozessschritte Analyse, Konzeption, Umsetzung/ Design, Evaluierung und Optimierung." (Vgl. [11])

Im Rahmen der Analyse werden die Anforderungen an das System erstellt und zusammengeführt. Hierzu können Personas erstellt werden (siehe: 4.1.1) und aus eben diesen können Anforderungen erstellt werden. Die Analyse soll zudem "Usability" [14] Ziele festlegen, anhand welcher in der "Evaluierung und Optimierung" die Güte der Nutzeroberfläche gemessen werden kann [11]. Während der Konzeption soll das Verständnis der Benutzer und deren Bedürfnisse hinsichtlich der User Experience auf die Benutzeroberfläche übertragen

werden [11, (ebd.)]. Die Konzeption endet mit dem erstellen eines Prototypen. Dieser wird in der Dritten Phase (Design) durch "ein konsistentes, ansprechendes und klares Grafik Design"(Vgl.) [11]) erweitert, was Probleme löst und eine Intuitive Bedienung unterstützt. Die Phase der "Evaluierung und Optimierung" wird das Produkt auf Probleme wie Sicherheit oder eine schlechte Bedienbarkeit hin untersucht. Hierdurch werden Mängel entdeckt welche in den nächsten Zyklus der Entwicklung einfließen.

Wie hoch der Grad der Usability und der User-Experience ist, lässt sich wie zuvor erwähnt anhand von Personas ableiten. Im Folgenden wird daher erklärt, was Personas sind und es werden eigene Personas für das Projekt erstellt.

4.1.1 Personas

Bei Personas handelt es sich nun um fiktive Personen, also hypothetische User, mit individuellen Eigenschaften [9,], welche ebenso eine reale Person haben könnte. Je ähnlicher die Personas also der Zielgruppen des Unternehmens/des Produktes sind, desto effektiver ist die Verwendung dieser.

Personas sind nicht nur als potentielle Zielgruppe zu sehen - sie haben zudem die Aufgabe, dass sich das Entwicklerteam in die Position des Nutzers versetzen kann. Dazu werden Personas zunächst nach Kategorien wie "Geschlecht" oder "Alter" aufgeteilt und anschließend wird jedes Profil mit Merkmalen und Eigenschaften gefüllt. Welche Informationen unter anderem zu Personas hinzugefügt werden, zeigt die folgende Tabelle.

Vor- und Nachname	Geburtsdatum / Alter
Foto der Person/Aussehen	Herkunft/Wohnort
Sprachkenntnisse	Beruf; Berufserfahrung in Jahren
Bildung/Ausbildung	Familienstand
Interessen und Hobbys	Fähigkeiten/Behinderungen
Sicherheitsrisikofaktoren für den Straßenverkehr	Abneigungen
Erfahrungen mit Technik (Meidenkompetenz)	Vorlieben
Fahrerfahrung	Kaufbereitschaft
Auto; Wenn ja: Besitzverhältnis?	

Anhand dieser möglichen Informationen werden im folgenden die Personas für dieses Projekt erstellt.

Name / Soziographische Daten	Hobbys & Interessen	Anwendungsfallbezogenes
 <ul style="list-style-type: none"> • Dietmar Müller, 68 Jahre (Bild: [10]) • Loppersum, Ostfriesland • Deutsch • Gelernter Maurer • Verheiratet, 3 Erwachsene Kinder 	<ul style="list-style-type: none"> • Rentner; Taxiunternehmer(seit 20 Jahren aktiv) • Fischen, Wandern • Handwerklich begabt, Hobbygärtner • Diabetiker • idR. Minus-Kunde • Geizig 	<ul style="list-style-type: none"> • Führerschein seit 47 Jahren • Vertraut Technik nicht • Schlechte Sehkraft • Ängstlicher Autofahrer • Besitzt eigenen Neuwagen
 <ul style="list-style-type: none"> • Jake Schneiders, 52 Jahre (Bild: [3]) • Washington, USA • Englisch, Russisch(Fließend) • Ausgebildeter Cyberhacker • Geschieden, ein Kind (geteiltes Sorgerecht) 	<ul style="list-style-type: none"> • Leutnant beim Militär • Jagen, Baseball Trainer • Kindersorgerecht unter der Woche • Workaholic • Chancen-Kunde 	<ul style="list-style-type: none"> • Führerschein seit 27 Jahren • Vertraut sehr auf Technik • guter, sicherer Autofahrer • Setzt Wert auf gute Bedienbarkeit • Hat gerne die Kontrolle über Systeme • Fährt einen Wagen des Militärs • Arbeitet viel am Laptop

 <ul style="list-style-type: none"> • Chi Nguyen, 24 Jahre (Bild: [13]) • Rom, Italien • Vietnamesisch, Italienisch(Fließend), Englisch (Fließend) • Studentin (Geschichte) • Single 	<ul style="list-style-type: none"> • Studentin, Stadtführerin in Rom • Joggen, Roadtrips • Influencer • Plus-Kundin 	<ul style="list-style-type: none"> • Führerschein seit 2 Jahren • Vertraut sehr auf Technik • unsichere Auto-fahrerin • Ist mit viel Technik aufgewachsen • Mag es, ihre Apps zu personalisieren • Nimmt Teil an Car-Sharing
--	---	--

Die erstellten Personas werden bei der Erstellung des Prototypen herangezogen um Anforderungen an das System. Jede einzelne ermöglicht es, bestimmte Schwierigkeiten und Herausforderungen genauer darzustellen. So soll Dietmar Müller verdeutlichen, dass ein Kunde möglicherweise doch vom Kauf einer SW überzeugt werden kann, obwohl er idR. ein Minus-Kunde ist. Durch ihn kann auch dargestellt werden, welche weiteren Einflussfaktoren es neben "Stress" o.Ä. gibt, wie bei seine Diabetes Erkrankung.

Jake Schneiders soll die Art Person darstellen, welche neugierig ist, die Sicherheit eines System testen möchte und hierzu auch selber in der Lage ist. Durch seine technische Affinität kann er Systeme leicht Verstehen und neue Anforderungen erstellen, die voraussichtlich leicht zu implementieren sind.

Chi Nguyen ist eine sehr feminine Studentin welche bereits vielerorts gewohnt hat. Durch ihr Dasein als Influencerin hat sie ein großes Netzwerk an Followern, mit welchen sie ihre Erfahrungen die sie während der Autofahrt sammelt Teilen kann.

Da alle Personas in unterschiedlichen Ländern an dem Straßenverkehr teilnehmen, sind die Einflussfaktoren zur Erstellung von Anforderungen sehr vielfältig, was sich positiv auf die Entwicklung auswirken kann.

4.2 Tech-Scouting

Damit die Entwicklung des Prototypen möglichst simpel sein wird, wird im Folgenden ein Technologie-Scouting durchgeführt. Es soll die Vorteile aufzeigen, welche ausgewählte En-

twicklungsumgebungen, Programmiersprachen und Frameworks für die Entwicklung des Prototypen haben.

Im Rahmen eines Prototypen für die Erkennung und Bereitstellung neuer Fahrfunktionen für intelligente Fahrzeuge ist es nötig, ein auf der Straße fahrendes Fahrzeug zu Simulieren und dazu eine passende Nutzeroberfläche zu haben, welche als Mensch-Maschine Schnittstelle des Fahrzeugs agiert. Für das Simulieren einer Straßenverkehrssituation wird der OpenSource Simulator **Carla**⁷ verwendet. Mit Carla ist es möglich, das eigene Auto zusammen mit anderen Fahrzeugen, Radfahrern, Fußgängern, Hindernissen uvm. auf der Straße darzustellen. Autos können mit Sensoren ausgestattet werden und die aufgenommenen Daten können mittels der Python-API ausgelesen werden. Der Ablauf einer Simulation wird in einem Python-Script spezifiziert. Eine alternative Möglichkeit zur Erstellung einer Simulation ist das einbinden einer OpenSCENARIO-Datei. Es existiert ein Aufsatz auf Carla, wodurch der Simulator den in einer solchen Datei dargestellten Ablauf selber darstellen kann.

Für die Erstellung von Python Skripts wird "**PyCharm**"⁸ als Entwicklungsumgebung gewählt. PyCharm ermöglicht eine intuitive und einfache Anbindung an GIT. Da die aus Carla ausgelesenen Daten verarbeitet werden müssen um zu erkennen wann das Auto nicht mehr selbstständig fahren kann, ist die Entwicklung eines Servers notwendig. Neben der Analyse der Simulation soll dieser als Kommunikationszentrale zwischen eben diesem und der Mensch-Maschine-Schnittstelle dienen. Zur Entwicklung des Servers soll das ebenfalls von JetBrains entwickelte **IntelliJ IDEA**⁹ dienen. Auch IntelliJ bietet eine einfache Anbindung an GIT an - zusätzlich ist die Einbindbarkeit von **Maven**¹⁰ zum verwalten von *Dependencies* ein leichtes.

Für die Entwicklung der Mensch-Maschine-Schnittstelle wird eine Android App entwickelt - für diese wird **Android Studio**¹¹ verwendet. Android Studio ermöglicht neben der visuellen auch eine textuelle Bearbeitung von Nutzeroberflächen. Es ist die von Google empfohlene Entwicklungsumgebung und ermöglicht das erstellen von Apps massiv.

Neben Python wird zum entwickeln von Server und MMS hauptsächlich die Programmiersprache Java (Version 1.10) verwendet. Java wurde an der Universität Oldenburg gelehrt und es wird unter anderem zum entwickeln von Android Apps verwendet.

⁷ <http://carla.org/>

⁸ <https://www.jetbrains.com/de-de/pycharm/>

⁹ <https://www.jetbrains.com/de-de/idea/>

¹⁰ <https://maven.apache.org/>

¹¹ <https://developer.android.com/studio>

5 Ausblick und erstes Konzept der praktischen Umsetzung

Nachdem in den vorherigen Kapiteln einzelne Aspekte der Wertschöpfungskette erläutert wurden, soll es abschließend einen Ausblick auf die Bachelorarbeit geben und ein erstes technisches Konzept des Prototypen vorgestellt werden. Im Laufe der Arbeit haben sich einige zu erläuternde Aspekte aufgetan, welche aus Platzgründen erst in der Bachelorarbeit behandelt werden.

Im Rahmen dieser soll zunächst ein erster Zyklus des UCD durchgeführt werden, um so die Nutzeroberfläche der Mensch-Maschinen-Schnittstelle zu erarbeiten. Dabei sollen zusätzlich mögliche Personalisierungsfeature benannt werden. Weiterhin sollen die anderen im Folgenden vorgestellten Teilsysteme entwickelt werden. Unter anderem soll ein Suchalgorithmus für den SUPR des Servers erarbeitet werden sowie eine Gegenüberstellung von Rechenleistung und Sendeleistung des Autos bzw. des Servers erfolgen. Es werden die einzelnen Bausteine einer Wertschöpfungskette (zur Erkennung und Bereitstellung neuer Fahrumfänge intelligenter Fahrzeug) benannt, erläutert und im Prototypen veranschaulicht. Im Folgenden wird für diesen ein erstes technisches Konzept vorgestellt.

Erstes technisches Konzept des Prototypen

Abbildung 6 zeigt eine erste Architektur des Prototypen. Dieser ist unterteilt in das **Fahrzeug**(unten) und den **Server**(oben). Das Fahrzeug ist in drei Teilsysteme aufgeteilt. Der "Carla Python Skript Client" ist das Modul in dem die Simulation stattfindet. Entscheidungen bzgl. der Fahraufgabe werden in diesem getroffen. Die **Mensch-Maschine-Schnittstelle** ist eine Android-basierte App. Sie wird über einen Emulator bedienbar sein und soll unter anderem Informationen aus dem Carla Simulator darstellen. Damit dies möglich ist, wird der dritte Baustein des Fahrzeugs eingeführt: der SUPR.

Der SUPR des Fahrzeugs soll die im Forschungsseminar definierten Aufgaben durchführen können und dient zusätzlich als Kommunikationseinheit. Er verbindet die Carla Simulation und die Mensch-Maschine-Schnittstelle, sodass eine interne Kommunikation im Fahrzeug möglich ist. Des Weiteren dient der SUPR als Kommunikationseinheit zum Server. Er ist für die Installation eingehender Softwarevorschläge zuständig und muss entsprechende Updates an den Python Client senden. In der Uptane Architektur waren hier die ECU's eines Autos angedacht. Da diese in dem ersten Konzept nicht auftauchen, steuert dies der SUPR.

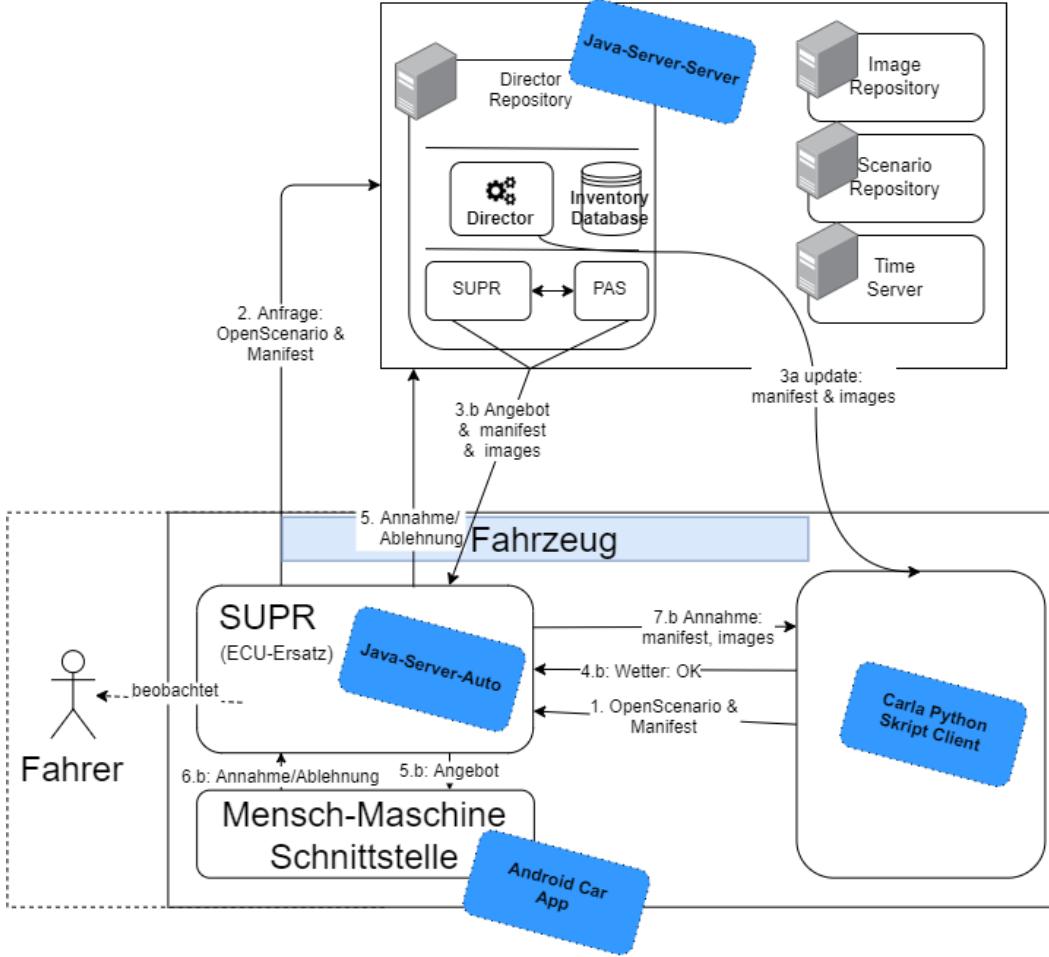


Figure 6: Technisches Erstkonzept

Die Architektur des Servers ist identisch zu der aus Kapitel 3.2. In der Implementierung sollen der Einfachheit halber die vier Bestandteile des Servers auf einem Server laufen und nicht getrennt werden. Der SUPR des Director Repositorys soll die im Forschungsseminar ihm zugeschriebenen Aufgaben durchführen können, gleiches gilt für den PAS. Die einzelnen Bestandteile des Servers können mit Ausnahme des Time-Servers miteinander kommunizieren.

Aus den Pfeilen des Konzepts abzuleiten ist der mögliche Ablauf eines Kaufprozesses. Initiiert wird dieser dadurch, dass das Fahrzeug im Carla Client nicht mehr autonom Fahren kann. Dieser erstellt infolgedessen eine OpenScenario-Datei und schickt diese gemeinsam mit dem Manifest, in welchem die bereits installierte Software notiert ist, an den SUPR (Pfeil 1). Dieser schickt den entdeckten Bedarf als Anfrage an den Server (Pfeil 2), welcher infolgedessen überprüft welche Software den entdeckten Bedarf abdecken kann. Hierzu sucht der Director des Servers im Manifest des Fahrzeugs nach zu aktualisierender Software. Wird ein Defizit festgestellt, stellt der Server eine Verbindung mit dem Python Client her, welcher

dieses Update installiert (Pfeil 3.a). Ist jede Software wieder aktuell stellt der SUPR des Servers fest, ob die Bedarfssituation mit den Updates bewältigt werden kann. Ist dies **nicht** der Fall, sucht der SUPR nach der passenden Software anhand der OpenScenario Datei. Ist ein Paket gefunden, wird das vom PAS erstellt Angebot zusammen mit dem unterschriebenen Manifest und dem Software-Image an den SUPR zurückgeschickt (Pfeil 3b).

Wird eine neue Software vorgeschlagen, hält der SUPR den Vorschlag so lange von der Nutzeroberfläche fern, wie der Fahrer sich noch in einer 'stressigen' Situation befindet. Ob dies der Fall ist, wird anhand des Wetters abgeleitet. Ist es regnerisch, nebelig oder ähnliches, soll das Angebot zurückgehalten werden. Bei sonnigem Wetter darf es dann auf der Nutzeroberfläche angezeigt werden. (Pfeil 4.b) Der Fahrer wird anhand einer Mitteilung darüber informiert, dass eine neue mögliche Software vorhanden ist. Der Fahrer kann das Angebot in der Nutzeroberfläche anpassen und es abschließend entweder annehmen oder ablehnen(Pfeil 6.b). Wird das Angebot angenommen, installiert der SUPR (Java-Client) die Software auf dem Fahrzeug, indem er eine Nachricht an das Python Skript schickt. Die Simulation wird erneut gestartet und das Auto kann die zuvor schwierige Situation bewältigen.

Abbildungen

1	Architektur AV nach Jeff Schneider (CMU) [5]	3
2	Ausschnitt einer OpenSENARIO Datei von Andreas Biehn [8, (Downloads)]	4
3	Umweltanalyse	7
4	Architektur Design UPTANE [7]	8
5	Anangepasste Uptane Architektur	10
6	Technisches Erstkonzept	23

Quellenverzeichnis

- [1] (Aufgerufen am 10. Januar 2020) Andrea Rosenbusch. User-centered design in sieben punkten kurz erklärt. https://zeix.com/wp-content/uploads/2011/04/ict-jb2011_artikel_zeix_rosenbusch_felix_final.pdf, 2011.
- [2] (Aufgerufen am 10. Januar 2020) Divya Krishnan. Ui/ux for autonomous vehicle interface to build trust. <https://medium.com/divya-krishnan-design/ui-ux-for-autonomous-vehicle-interface-to-build-trust-de7f4c545c3b>, 2019.
- [3] (Aufgerufen am 10. Januar 2020) GrummelJS. Datei:walter e. boomer.jpg. https://de.wikipedia.org/wiki/Datei:Walter_E._Boomer.jpg, 2008.

- [4] (Aufgerufen am 10. Januar 2020) Hans-Christian Reuss. Intelligenten fahrzeuge. https://www.uni-stuttgart.de/presse/archiv/themenheft/07/intelligente_fahrzeuge.pdf, 2010.
- [5] (Aufgerufen am 10. Januar 2020) Jeff Schneider. Ri seminar: Jeff schneider : Self driving cars and ai. https://www.youtube.com/watch?v=jTio_MPQRyC&t=1497s, 2019.
- [6] (Aufgerufen am 10. Januar 2020) Juergen Daunis. The automotive industry in transformation – business model disruption. <https://www.ericsson.com/en/blog/2017/11/the-automotive-industry-in-transformation--business-model-disruption>.
- [7] (Aufgerufen am 10. Januar 2020) Linux Foundation. Uptane design. <https://uptane.github.io/design.html>, 2020.
- [8] (Aufgerufen am 10. Januar 2020) Marius Dupius. Open scenario. <https://www.asam.net/standards/detail/openscenario/>, 2015.
- [9] (Aufgerufen am 10. Januar 2020) Martin Seibert. Personas geben zielgruppen gesichter. <https://blog.seibert-media.net/blog/2008/09/12/personas-geben-zielgruppen-gesichter/>, 2008.
- [10] (Aufgerufen am 10. Januar 2020) Michael Green. Angry granpa. [https://commons.wikimedia.org/wiki/File:Angry_Grandpa_-_2015_\(cropped\).jpg](https://commons.wikimedia.org/wiki/File:Angry_Grandpa_-_2015_(cropped).jpg), 2015.
- [11] (Aufgerufen am 10. Januar 2020) Seobility. User centered design.
- [12] (Aufgerufen am 10. Januar 2020) Thomas Weißgerber. Konzepte und methoden des user centered design. http://www.inf.fu-berlin.de/inst/ag-se/teaching/S-BSE/326_weissgerber_UCD.pdf, 2015.
- [13] (Aufgerufen am 10. Januar 2020) Tibchris. File:a beautiful girl in her black dress.jpg. https://commons.wikimedia.org/wiki/File:A_beautiful_girl_in_her_black_dress.jpg, 2010.
- [14] (Aufgerufen am 10. Januar 2020) usability toolkit.de. usability-toolkit.de: Usability für webprojekte. <http://usability-toolkit.de/usability/usability-in-web-projekten/>, 2020.
- [15] William Sims Bainbridge. *Berkshire encyclopedia of human-computer interaction*, volume 1. Berkshire Publishing Group LLC, 2004.
- [16] Verband der Automobilindustrie. Automatisierung - von fahrassistentensystemen zum automatisierten fahren. *VDA Magazin - Automatisierung*, 2015.
- [17] Lex Fridman. 'human-centered autonomous vehicle systems: Principles of effective shared autonomy'. Technical report, Massachusetts Institute of Technology (MIT), 2018.
- [18] Markus Reinke. *30 Minuten Verkaufsprychologie*. Gabel Verlag, 2013.

- [19] R. Pokam-S. Langlois S. Debernard, C. Chauvin. Designing human-machine interface for autonomous vehicles. Technical report, International Federation of Automatic Control, 2016.
- [20] Gerd-Inno Spindler. *Kaufverhalten und Kaufentscheidung*, pages 39–48. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.
- [21] Niklas Stelter. Entwurf und simulation von monitoren für die evaluierung von updates intelligenter fahrzeuge. Master's thesis, CvO Universität Oldenburg, 2019.

Versicherung Eides Statt

Hiermit versichere ich,

Linus Hestermeyer, wohnhaft in Marschweg 92 in 26131 Oldenburg, Matr.-Nr.: 4087097,

an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, 10. Januar 2020

Ort, Datum

Unterschrift